Tutorial and workshop „Modelling with GroIMP and XL“ / Tutorial for beginners

University of Göttingen, 27 February, 2012

Winfried Kurth

Basic examples in XL (part 1)

related material: XL code files **`sm09_e??.rgg`**
URL **`http://www.uni-forst.gwdg.de/~wkurth/`**
and drive **`T:\rgg`** on CIP pool stations

Remember:

L-systems (Lindenmayer systems)

rule systems for the replacement of character strings

in each derivation step *parallel* replacement of all characters for which there is one applicable rule

interpreted L-systems:

turtle command language is subset of the alphabet of the L-system

Aristid Lindenmayer (1925-1989)

Example:

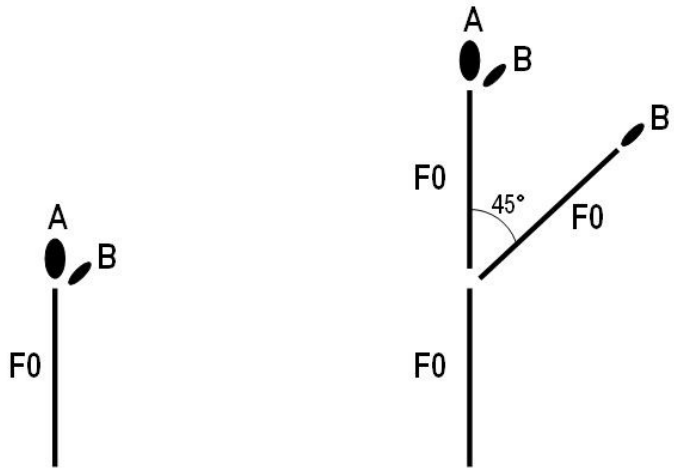
rules

```
A ==> F0 [ RU(45) B ] A ;
B ==> F0 B ;
```

start word **A**

A → F0 [RU(45) B] A → F0 [RU(45) F0 B] F0 [RU(45) B] A → ...

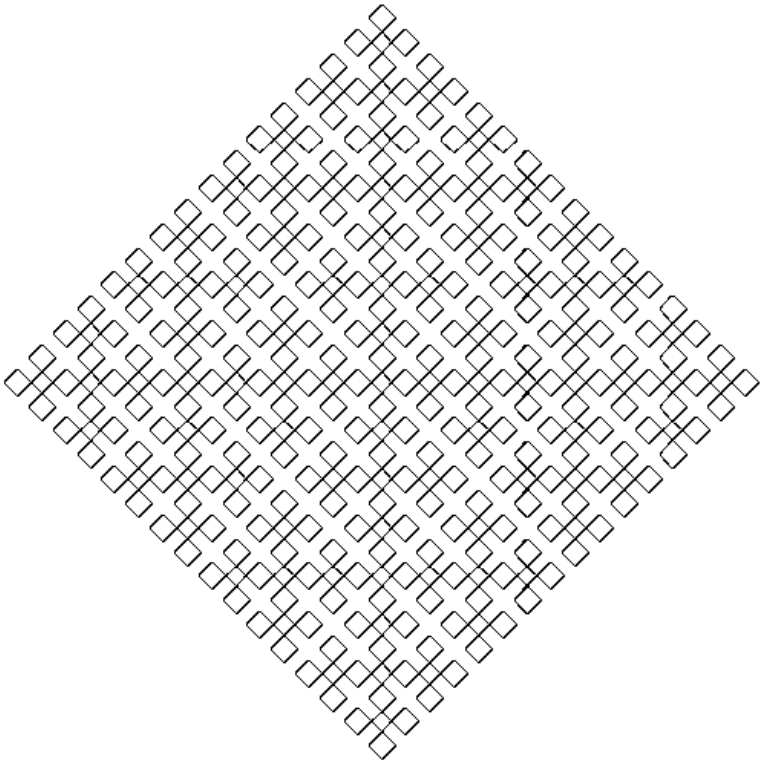
interpretation
by
turtle geometry



(**A** and **B** are normally not interpreted geometrically.)

example space filling curve:

```
Axiom ==> L(10) RU(-45) X RU(-45) F(1) RU(-45) X;
X ==> X F0 X RU(-45) F(1) RU(-45) X F0 X
```

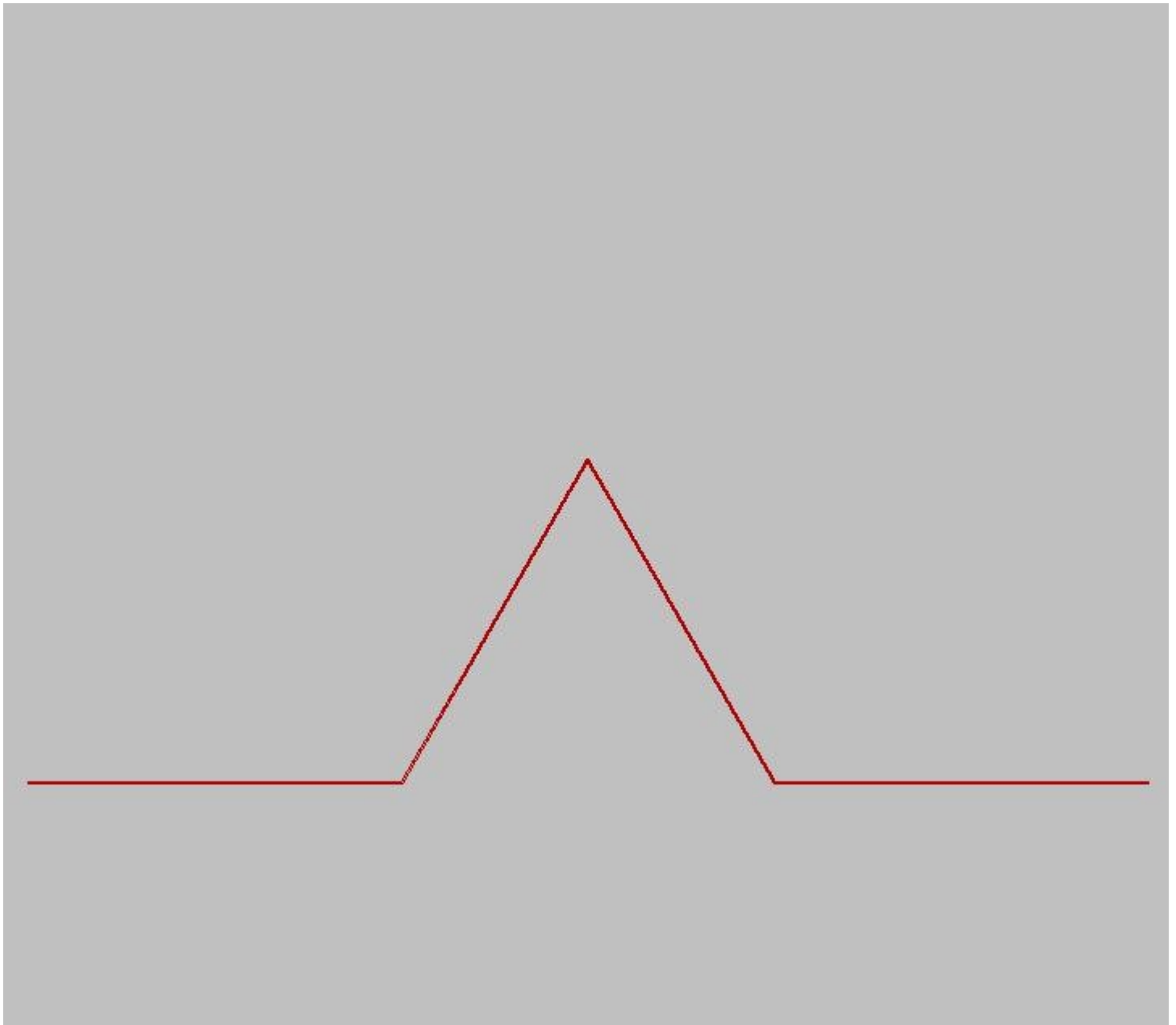


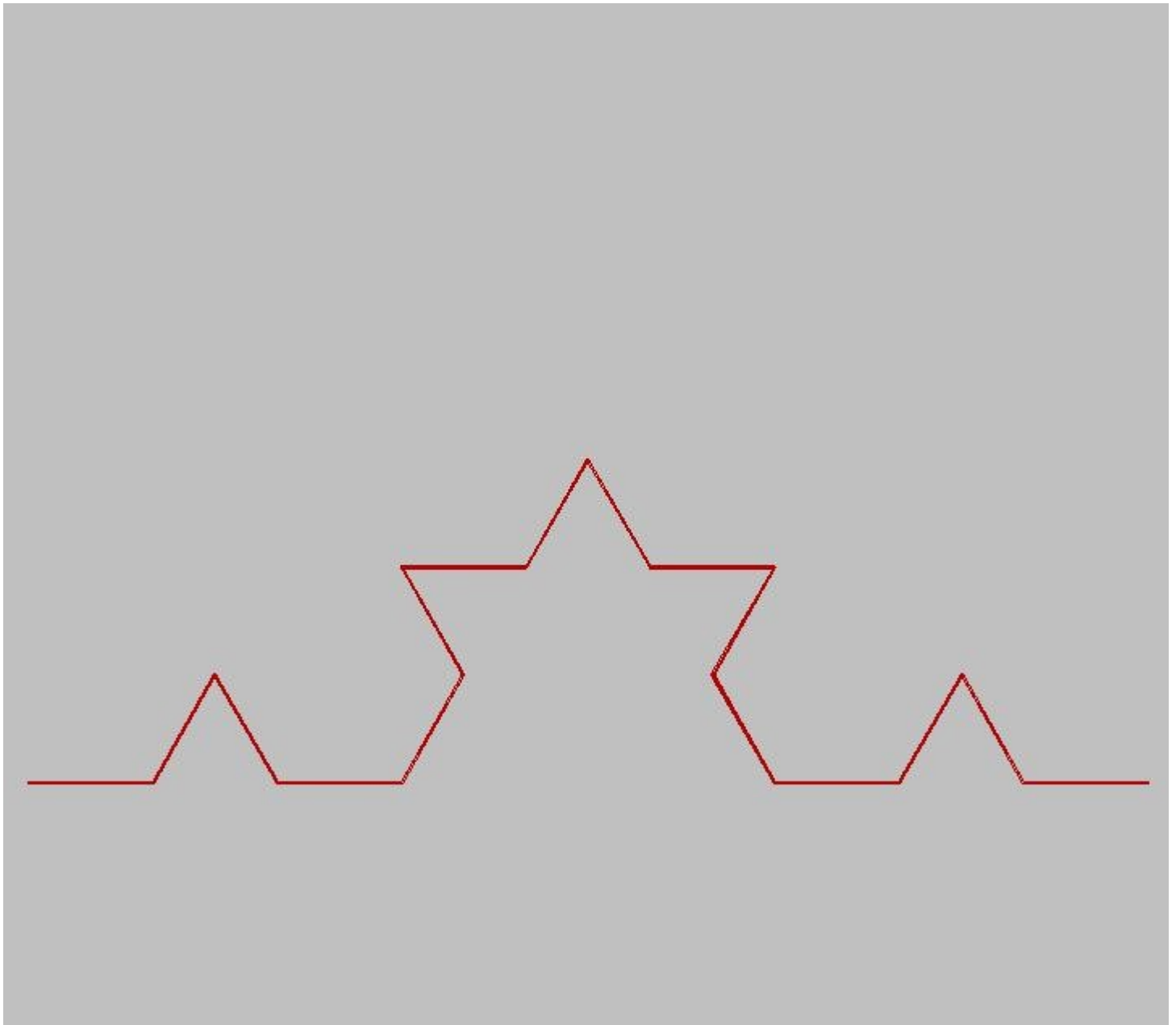
traditional Indian kolam
„Anklets of Krishna“

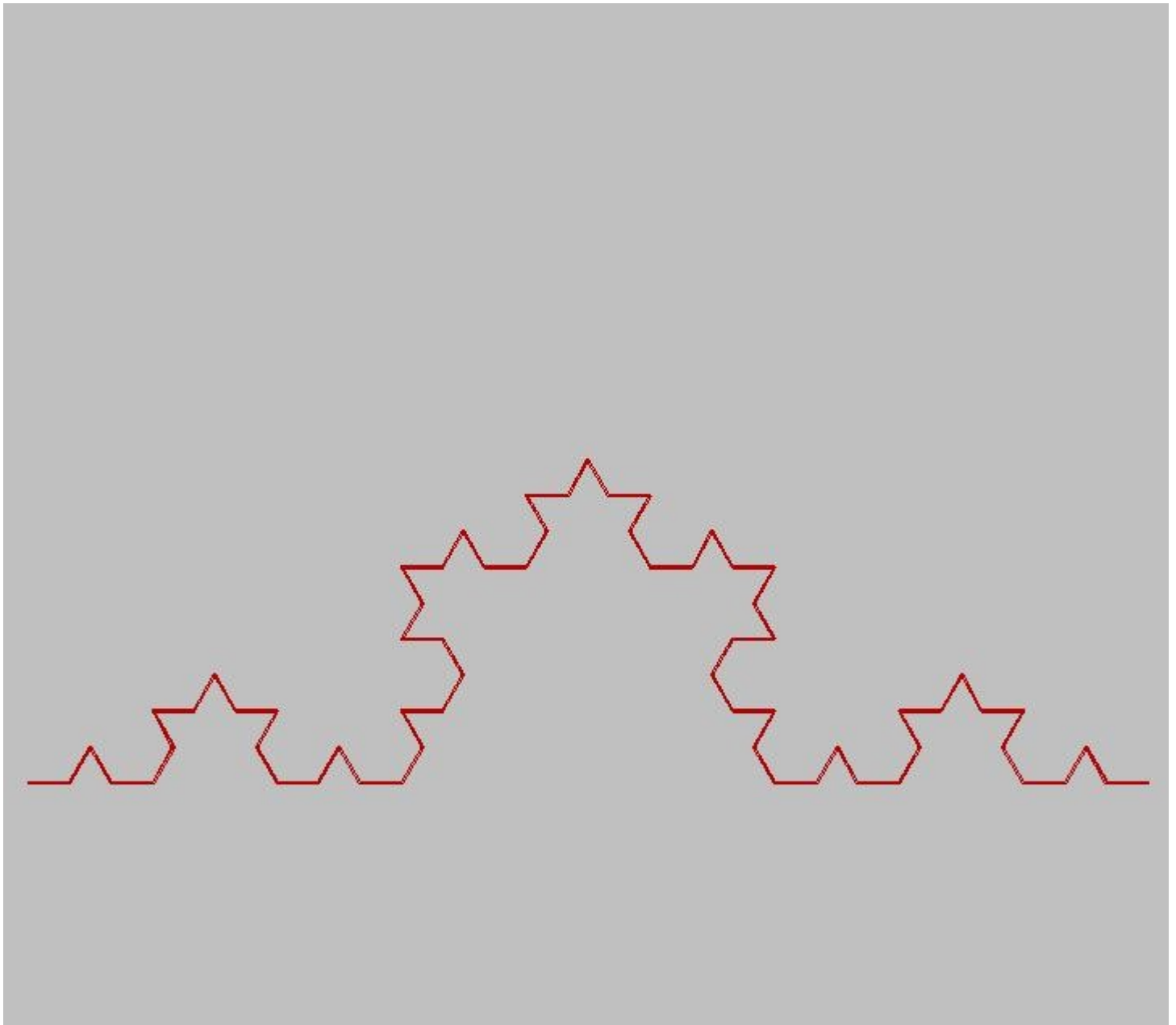
example for a fractal:

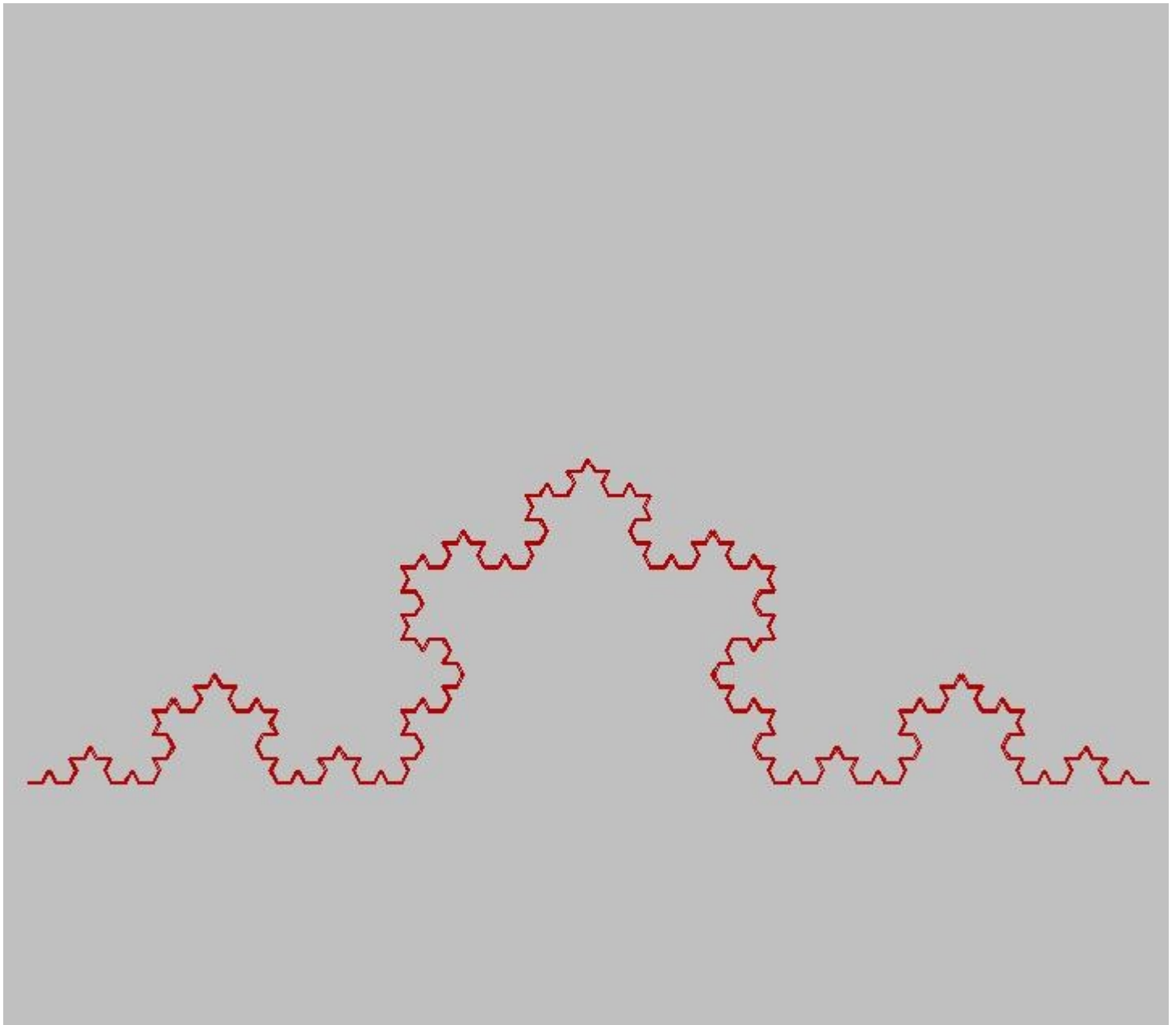
Koch curve

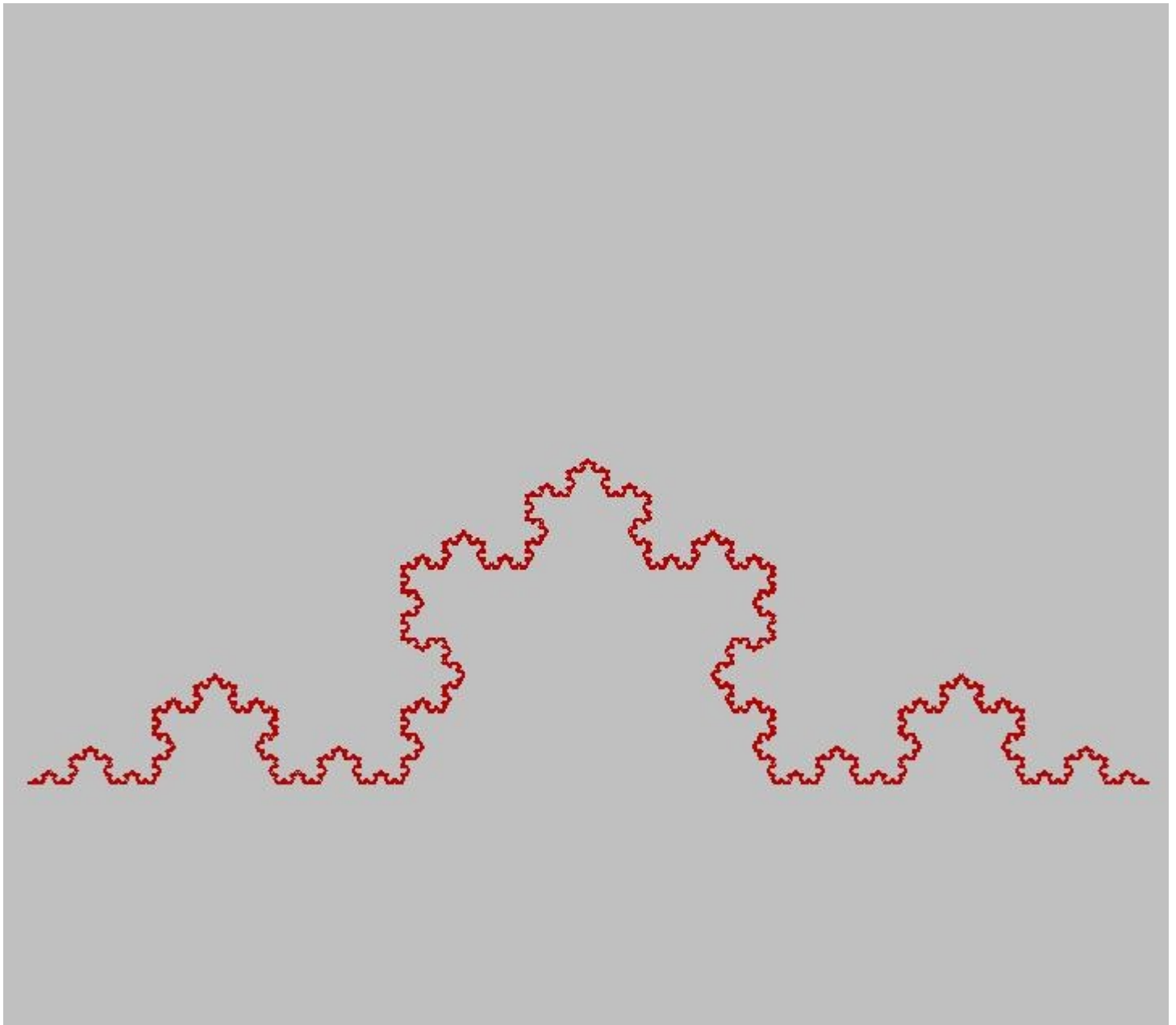
```
Axiom ==> RU(90) F(10);
F(x) ==> F(x/3) RU(-60) F(x/3) RU(120) F(x/3) RU(-60) F(x/3)
```











cf. sample file **sm09_e02.rgg** :

closed Koch curve, developed from triangle

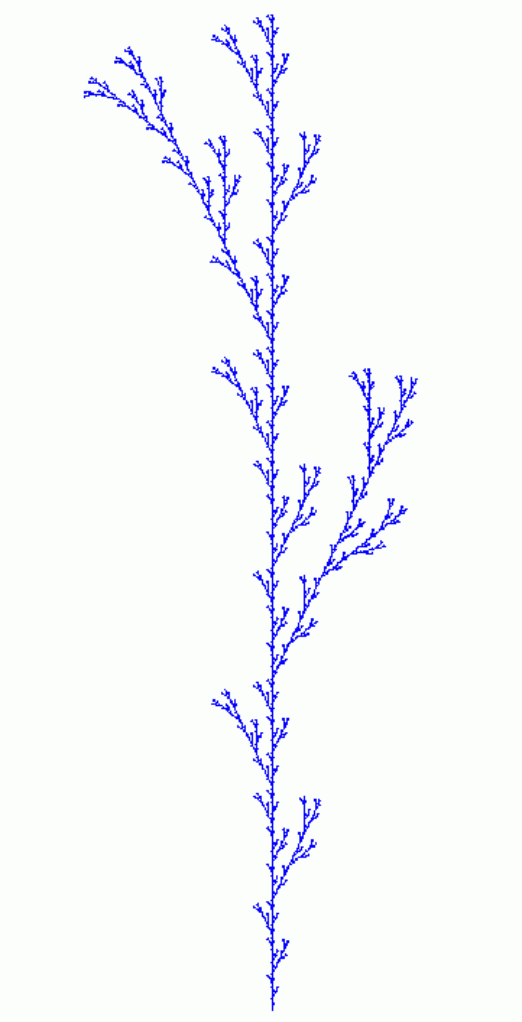
```
protected void init()
[ Axiom ==> RU(50) F(10) RU(120) F(10) RU(120) F(10); ]

// public method for interactive usage in GroIMP
// (via button):
public void application()
// rules must be set in [] and finished with ;
[
        // each F() is replaced by 4 smaller F()
        // the length of the F on the left-hand side is taken over
        // by x to the right-hand side

        F(x) ==> F(x/3) RU(-60) F(x/3) RU(120) F(x/3) RU(-60) F(x/3);
]
```

example with branching:

```
F0 ==> F0 [ RU(25.7) F0 ] F0 [ RU(-25.7) F0 ] F0 ;
```

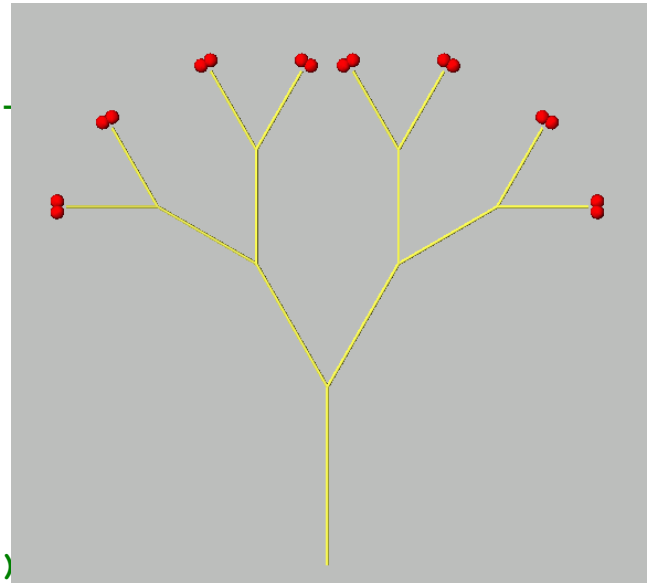
Result after 7 steps:

(start word `L(10) F0`)



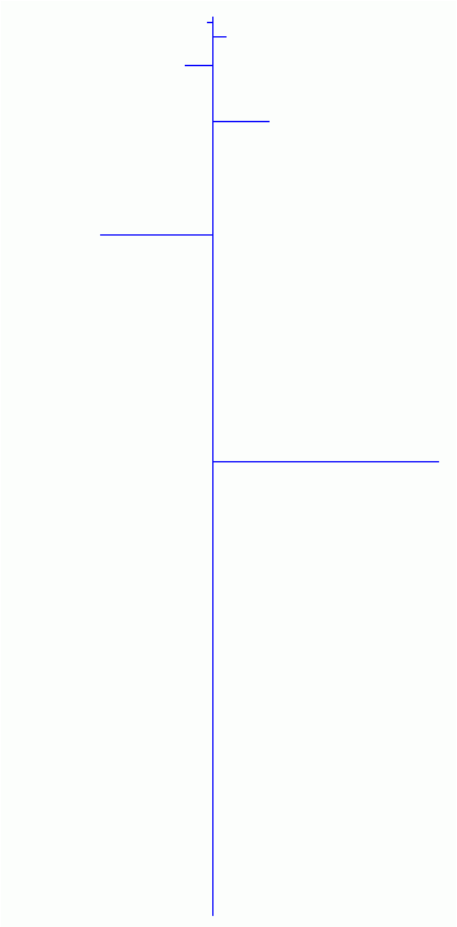
sample file sm09_e03.rgg :

```
/* You learn at this example:
- how to construct a simple plant model (according to architectural model Schoute)
- how to specify branches with [ ]  */

// Example of a simple tree architecture (Schoute architecture)

//----------- Extensions to the standard alphabet ----------
//Shoot() is an extension of the turtle-command F() and stands for an annual shoot
module Shoot(float len) extends F(len);

// Bud is an extension of a sphere object and stands for a terminal bud
// its strength controls the length of the produced shoot in the next timestep
module Bud(float strength) extends Sphere(0.2)
{{ setShader(RED); setTransform(0, 0, 0.3); }};
//-----------------------------------------------------

protected void init ()
[  // start structure (a bud)
   Axiom ==> Bud(5);
]

public void run ()
[
   // a square bracket [] will indicate a branch
   // (daughter relation)
   // Rotation around upward axis (RU) and head axis (RH)
   // Decrease of strength of the Bud (each step by 20%)

   Bud(x) ==> Shoot(x) [ RU(30) Bud(0.8*x) ] [ RU(-30) Bud(0.8*x) ];
]
```



branching, alternating position and shortening:

```
Axiom ==> L(10) F0 A ;

A ==> LMul(0.5) [ RU(90) F0 ] F0 RH(180) A ;
```



in XL, **A** must be declared as module before:

```
module A;
```

extension of the concept of symbol:

allow real-valued parameters not only for turtle commands like "**`RU(45)`**" and "**`F(3)`**", but for all characters

→ *parametric L-systems*

arbitrarily long, finite lists of parameters
parameters get values when the rule matches

Example:

rule **`A(x, y) ==> F(7*x+10) B(y/2)`**

current symbol is e.g.: **`A(2, 6)`**
after rule application: **`F(24) B(3)`**

parameters can be checked in conditions
(logical conditions with Java syntax):

`A(x, y) (x >= 17 && y != 0) ==>` ....

Test the examples

`sm09_e04.rgg`	two blocks of rules
`sm09_e05.rgg`	alternating phyllotaxis of branches
`sm09_e06.rgg`	opposite phyllotaxis
`sm09_e07.rgg`	colour specification for single elements by imperative code
`sm09_e08.rgg`	usage of your own modules
`sm09_e21.rgg`	positioning of lateral branches

Usage of imperative code in XL programmes

Commands like the assignment of values to variables, additions, function calls, output (print commands) etc. are specified in the same way like in Java and enclosed in braces { ... } .

Examples:

```
int i;   // declaration of an integer variable with name i

float a = 0.0; // declaration and initialization of a floating-point var.

int[] x = new int[20]; // declaration of an array
                       // of length 20; access: x[0], ..., x[19]
float[] y = { 0.1, 0.2, 0.7, -1.4 };
                       // declaration and initialization of an array

i = 25;  // assignment

i++;     // i is incremented by 1

i--;     // i is decremented by 1
```

usage of imperative code (continued)

```
i += 5;                 // i is incremented by 5
i -= 5;                 // i is decremented by 5
i *= 2;                 // i is doubled
i /= 3;                 // i gets the value i/3
n = m % a;      // n gets assigned the rest of m from integer division by a
x = Math.sqrt(2);               // x gets assigned the square root of 2
if (x != 0) { y = 1/x; }  // conditional assignment of 1/x to y
while (i <= 10) { i++; }  // loop: as long as i ≤ 10,
                                // i is incremented by 1
for (i = 0; i < 100; i++) { x[i] = 2*i; }  // imperative
                                                // for-loop
if (i == 0) { ... }       // test for equality ( „=“ would be assignment!)
```

data types:

`int`	integers
`float`	floating-point numbers
`double`	floating-point numbers, double precision
`char`	characters
`void`	void type (for functions which return no value)

mathematical constants:

`Math.PI`	π
`Math.E`	*e*

logical operators:

`&&`	and
`||`	or
`!`	not

mathematical functions:

`Math.abs`	absolute value
`Math.acos`	arcus cosine
`Math.asin`	arcus sine
`Math.atan`	arcus tangens
`Math.cos`	cosine
`Math.exp`	exponential function e^x
`Math.log`	natural logarithm
`Math.max`	maximum of two numbers
`Math.min`	minimum of two numbers
`Math.round`	functin for rounding
`Math.sin`	sine
`Math.sqrt`	square root
`Math.tan`	tangens
`Math.toDegrees`	
`Math.toRadians`	conversion degrees $\leftrightarrow$ radians

sm_progbsp01.rgg: writes the numbers from 1 to 10 to the GroIMP console

```
protected void init()
{
  int i;
  for (i=1; i<= 10; i++)
     println(i);
  println("end.");
}
```

sm_progbsp02.rgg: writes odd square numbers

```
protected void init()
{
   int a, b;
   for (a = 1; a <= 10; a++)
     {
     b = a*a;
     if (b % 2 != 0) println(b);
     }
   println("end.");
}
```

sm_progbsp03.rgg: writes the Fibonacci numbers

```
protected void init()
{
    int i;
    int[] fibo = new int[20]; /* array declaration */
    fibo[0] = fibo[1] = 1;
    for (i=2; i <= 19; i++)
       fibo[i] = fibo[i-1] + fibo[i-2];
    for (i=0; i <= 19; i++)
       println(fibo[i]);
    println("end.");
}
```

`sm_progbsp04.rgg`: Usage of a function

```
/* a simple imperative programme:
A function written by the user calculates x² + 1;
this is evaluated for x from 0 to 1 in steps by 0.1.
Be aware of rounding errors and of the correct upper limit for x. */

public float function(float x)
    {
    return x*x + 1;
    }
protected void init()
    {
    float a = 0.0;              /* floating point number */
    while (a <= 1.00001)
        {
        println(function(a)); /* apply function and print */
        a += 0.1;             /* increment a */
        }
    println("end.");
    }
```

test the examples

`sm09_e20.rgg` usage of arrays

`sm09_e22.rgg` for-loop for lateral branches

Remember:

parameters can be checked in conditions
(logical conditions with Java syntax):

```
A(x, y) (x >= 17 && y != 0) ==> ....
```

test the examples

`sm09_e11.rgg`	conditions for rule applications
`sm09_e12.rgg`	conditions for rule applications (second variant)
`sm09_e13.rgg`	connection of two conditions

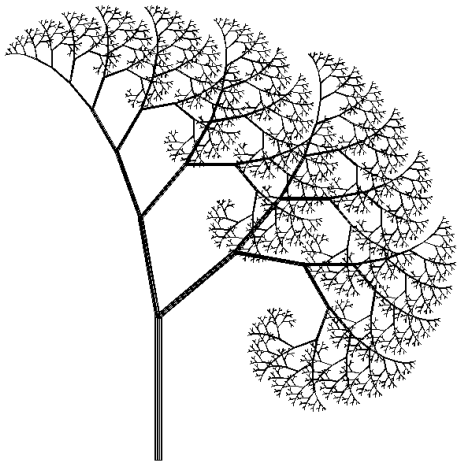
Stochastic L-systems

usage of pseudo-random numbers

Example:

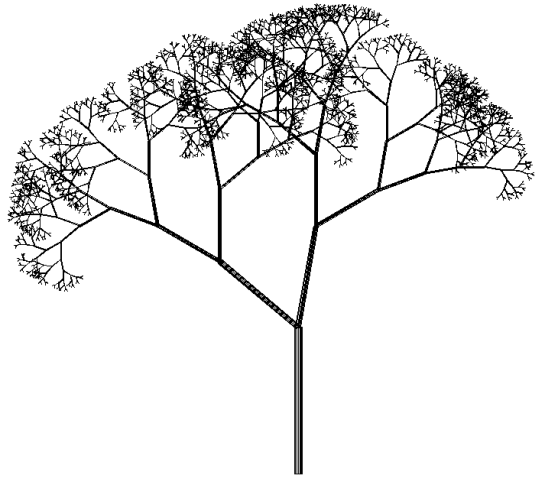
deterministic

```
Axiom ==> L(100) D(5) A;

A ==> F0 LMul(0.7) DMul(0.7)
   [ RU(50) A ] [ RU(-10) A ];
```



stochastic

```
Axiom ==> L(100) D(5) A;

A ==> F0 LMul(0.7) DMul(0.7)
   if (probability(0.5))
    ( [ RU(50) A ] [ RU(-10) A ] )
   else
    ( [ RU(-50) A ] [ RU(10) A ] );
```



XL functions for pseudo-random numbers:

`Math.random()`	generates floating-point random number between 0 and 1
`random(a, b)`	generates floating point random number between **`a`** and **`b`**
`probability(x)`	gives 1 with probability x, 0 with probability 1–x

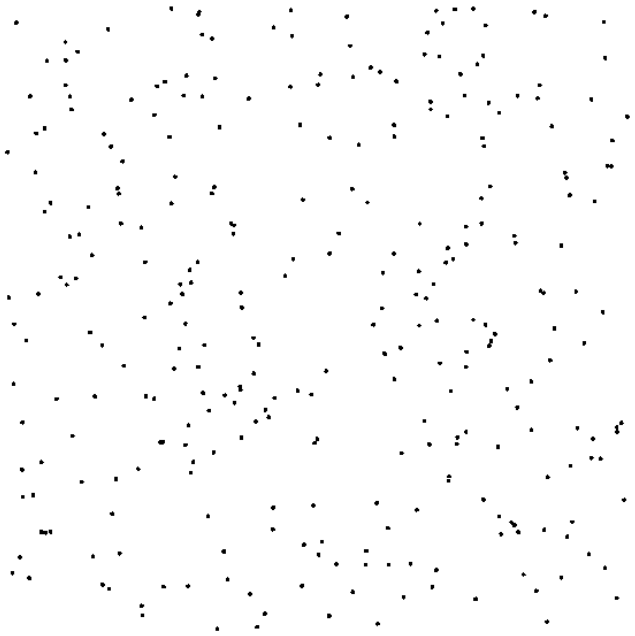
test the example

`sm09_b19.rgg` stochastic L-system

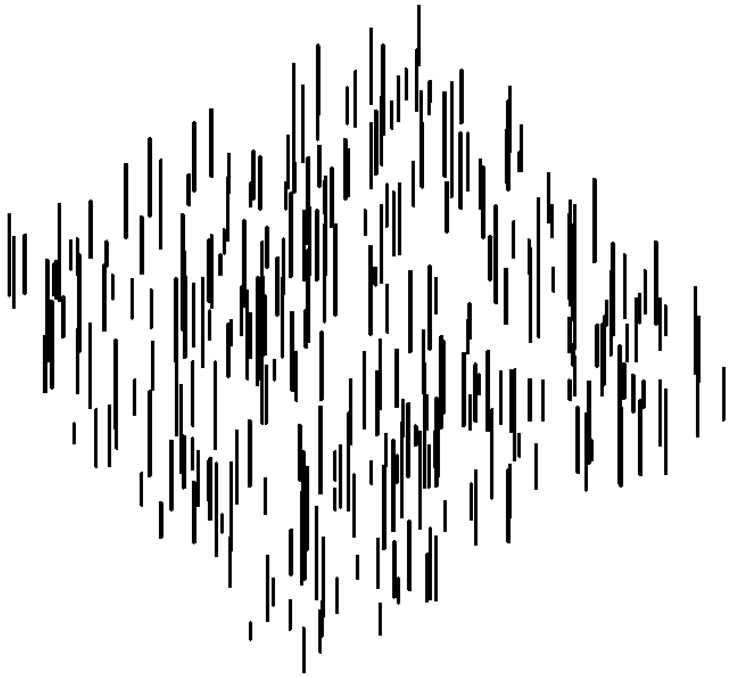
How to create a random distribution in the plane:

```
Axiom ==> D(0.5) for ((1:300))
              ( [ Translate(random(0, 100), random(0, 100), 0)
                  F(random(5, 30)) ] );
```

view from above



oblique view



How to create a GroIMP project with textures from graphics files (e.g., photos of leaves or bark)

1. File → New → RGG Project

2. insert name of the RGG file (text file)

3. delete the default programme from the GroIMP editor, write new programme or insert it from another source file

4. store file in the editor (automatic compilation must be successful)
   - textured objects are still shown in simplified form (without textures)

5. Panels → Explorers → 3D → Shaders → Object → New → Lambert

6. click twice on the name „Lambert“ (with delay between the clicks), overwrite it with the name which is foreseen in the programme (argument of the function „shader(...)“), finish with <return> (don‘ forget this!!)

7. doubleclick on sphere icon → Attribute Editor opens

8. click there on: Diffuse colour → Surface Maps → Image

9. click there on: Image [?] → From File

how to create a project
(continued)

10. choose image file, „open“
11. „Add the file“: OK
12. store editor file again / compile

    - textured objects are now shown with texture

13. to store the complete project:

    File $\rightarrow$ Save, write name of the project (must not necessarily coincide with the name of the RGG source code file).

test the example

`sm09_e10.gsz` usage of a surface texture (leaf texture)